

**compress** — Compress data in memory

[Description](#)  
Also see

[Quick start](#)[Menu](#)[Syntax](#)[Option](#)[Remarks and examples](#)

## Description

`compress` attempts to reduce the amount of memory used by your data.

## Quick start

Reduce the amount of memory used by the current dataset

```
compress
```

As above, but only reduce memory used by `v1` and `v2`

```
compress v1 v2
```

Speed up `compress` for large datasets with `strL`-type variables but possibly reduce the amount of memory saved

```
compress, nocoalesce
```

## Menu

Data > Data utilities > Optimize variable storage

## Syntax

```
compress [varlist] [, nocoalesce]
```

## Option

`nocoalesce` specifies that `compress` not try to find duplicate values within `stall` variables in an attempt to save memory. If `nocoalesce` is not specified, `compress` must sort the data by each `strL` variable, which can be time consuming in large datasets.

## Remarks and examples

[stata.com](#)

`compress` reduces the size of your dataset by considering two things. First, it considers demoting

<code>doubles</code>	to	<code>longs</code> , <code>ints</code> , or <code>bytes</code>
<code>floats</code>	to	<code>ints</code> or <code>bytes</code>
<code>longs</code>	to	<code>ints</code> or <code>bytes</code>
<code>ints</code>	to	<code>bytes</code>
<code>str#s</code>	to	shorter <code>str#s</code>
<code>strLs</code>	to	<code>str#s</code>

See [D] [data types](#) for an explanation of these storage types.

Second, it considers coalescing `strL`s within each `strL` variable. That is to say, if a `strL` variable takes on the same value in multiple observations, `compress` can link those values to a single memory location to save memory. To check for this, `compress` must sort the data on each `strL` variable. You can use the `nocoalesce` option to tell `compress` not to take the time to perform this check. If `compress` does check whether it can coalesce `strL` values, it will do whichever saves more memory—coalescing `strL` values or demoting a `strL` to a `str#`—or it will do nothing if it cannot save memory by changing a `strL`.

`compress` leaves your data logically unchanged but (probably) appreciably smaller. `compress` never makes a mistake, results in loss of precision, or hacks off strings.

### ▷ Example 1

If you do not specify a *varlist*, `compress` considers demoting all the variables in your dataset, so typing `compress` by itself is usually enough:

```
. use http://www.stata-press.com/data/r14/compmp2
(1978 Automobile Data)

. compress
variable mpg was float now byte
variable price was long now int
variable yenprice was double now long
variable weight was double now int
variable make was str26 now str17
(1,776 bytes saved)
```

If there are no compression possibilities, `compress` does nothing. For instance, typing `compress` again results in

```
. compress
(0 bytes saved)
```

◀

## Also see

[D] [data types](#) — Quick reference for data types

[D] [recast](#) — Change storage type of variable